ซอฟต์แวร์

สุทธิศักดิ์ พงศ์ธนาพาณิช
(e-mail : sutthisakp@yahoo.com)

เปิดโลกมัลติมีเดียด้วย Visual Basic ตอน Graphic Transformation สำหรับออบเจ็กต์ 2 มิติ (ตอนที่ 2)

วิธีการนำสมการคณิตศาสตร์เมตริกซ์ที่ใช้ในการ transformation จุดในระนาบ x-y มาเขียนเป็นโค้ดด้วย Visual Basic 5.0

จากตอนที่แล้วผู้เขียนได้อธิบายถึงทฤษฎีพื้นฐานด้านคณิตศาสตร์เมตริกซ์ที่เกี่ยวกับ Graphic Transformation สำหรับออบเจ็กต์ 2 มิติ โดยได้เน้นเฉพาะจุด (x, y) ในระนาบ x–y ทั้งนี้เพื่อให้ผู้อ่านที่ไม่คุ้นเคยกับคณิตศาสตร์เมตริกซ์สามารถทำความเข้าใจได้ง่าย สำหรับในบทความตอนนี้ผู้เขียนก็จะอธิบายโปรเจ็กต์ตัวอย่าง TransPnt.vbp ซึ่งจะแสดงวิธีการนำสมการคณิตศาสตร์เมตริกซ์ที่ใช้ในการ transformation จุดในระนาบ x–y มาเขียนเป็นโค้ดด้วย Visual Basic 5.0 ซึ่งในโปรเจ็กต์ตัวอย่างผู้อ่านจะได้เห็นวิธีการสร้างโพรซีเดอร์สำหรับการคำนวณด้านเมตริกซ์ และวิธีการแปลงระบบคู่ลำดับทั้งในรูปแบบของระบบคาร์ทีเซียน (cartesian) หรือระบบโพลาร์ (polar) ซึ่งเป็นระบบการให้ตำแหน่งคู่ลำดับที่ใช้กันอยู่ในปัจจุบัน และสำหรับผู้อ่านที่มีข้อสงสัยหรือต้องการพูดคุยกัน ก็สามารถส่งอี-เมล์มาได้ตามแอดเดรสดังนี้ครับ sutthisakp@yahoo.com

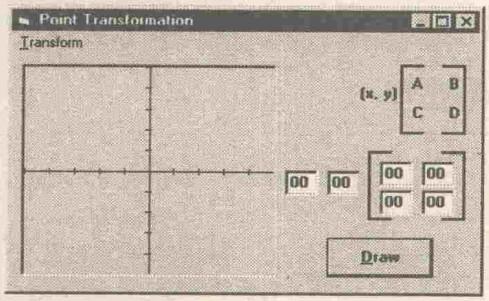
เนื่องจากในบทความตอนที่แล้วผู้อ่านได้อธิบายถึงทฤษฎีพื้นฐานเกี่ยวกับ Graphic Transformation เอาไว้อย่างละเอียด ดังนั้นในบทความตอนนี้ผู้เขียนจะไม่อธิบายซ้ำ แต่จะเน้นเฉพาะวิธีการเขียนโค้ดด้วย Visual Basic 5.0 จากสมการคณิตศาสตร์เท่านั้น และจะขอใช้เวลามากหน่อยในการอธิบายถึงวิธีการทำงานของโค้ดแต่ละขั้นตอนของโปรแกรม เพราะในบทความต่อ ๆ ไปที่ผู้เขียนจะกล่าวถึง Graphic Transformation สำหรับออบเจ็กต์ 2 หรือ 3 มิติอื่น ๆ นั้น ก็จะได้ไม่ต้องอธิบายถึงวิธีการทำงานของโค้ดแต่ละขั้นตอนของโปรแกรมมากนัก เพื่อจะได้มีเวลาอธิบายถึงสาระสำคัญด้านทฤษฎีและวิธีการประยุกต์ให้มากที่สุด ซึ่งคาดว่าน่าจะมีประโยชน์กับผู้อ่านสูงสุดครับ

การทำงานของโปรแกรม TransPnt

โปรแกรม TransPnt (โปรเจ็กต์ตัวอย่าง TransPnt.vbp) เป็นโปรแกรมตัวอย่างที่แสดงวิธีการเขียนโค้ดสำหรับการทำ Graphic Transformation กับจุดในระนาบ 2 มิติ (ระบบแกน x-y) ซึ่งจะเป็นโปรแกรมง่าย ๆ ที่ผู้อ่านเพียงแต่เลือกรายการ Graphic Transformation ที่ต้องการจากเมนู Transform จากนั้นก็คลิกที่ปุ่มคำสั่ง Draw เท่านั้น โปรแกรมก็จะทำการ transformation จุดตัวอย่าง (ค่า default ของตำแหน่งคู่ลำดับที่ผู้เขียนตั้งเอาไว้คือ x=1, y=2) ตามค่าที่ระบุในเมตริกซ์ ซึ่งผู้อ่านจะเห็นว่าวิธีการใช้งานโปรแกรมจะไม่ซับซ้อนเลย โดยเมื่อผู้อ่านรันโปรแกรม TransPnt จะปรากฏฟอร์มดังในรูปที่ 1 โดยในขณะนี้ผู้อ่านยังไม่สามารถทำการ transformation จุดโดยการคลิกที่ปุ่มคำสั่ง Draw เพราะในขณะที่เริ่มต้นรันโปรแกรมตำแหน่งคู่ลำดับจะมีค่าเท่ากับ (0, 0) ซึ่งการ transformation ด้วยเมตริกซ์ที่ผู้เขียนได้

นำเสนอในบทความตอนที่แล้ว จะไม่สามารถใช้ได้กับคู่ลำดับ (0, 0) ดังนั้นถ้าหากผู้อ่านคลิกที่ปุ่มคำสั่ง Draw ทันที โปรแกรมก็จะทำการวาดจุดที่ตำแหน่ง (0, 0) เท่านั้น

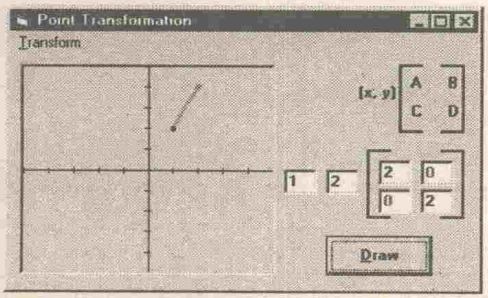
ด้วยเหตุนี้ก่อนที่ผู้อ่านจะสามารถคลิกที่ปุ่มคำสั่ง Draw ผู้อ่านต้องทำการเลือกรายการ transformation ที่ต้องการจากเมนู Transform เสียก่อน เพื่อให้โปรแกรมทำการกำหนดตำแหน่งคู่ลำดับที่ไม่ใช่ (0, 0) และกำหนดค่าตัวเลขสำหรับการ transformation จุดลงในเมตริกซ์เสียก่อน แล้วจึงค่อยคลิกที่ปุ่มคำสั่ง Draw เพื่อให้โปรแกรมทำการ transformation และแสดงผลตำแหน่งใหม่ของจุดคู่ลำดับหลังจากการ transformation อย่างถูกต้อง



รูปที่ 1 แสดงฟอร์มเริ่มต้นของโปรแกรม TransPnt

สำหรับการ transformation จุดในระนาบ x-y ที่โปรแกรมนี้สามารถทำได้มีอยู่ด้วยกัน 4 วิธี ดังที่ได้ระบุในคำสั่งของเมนู Transform ดังต่อไปนี้

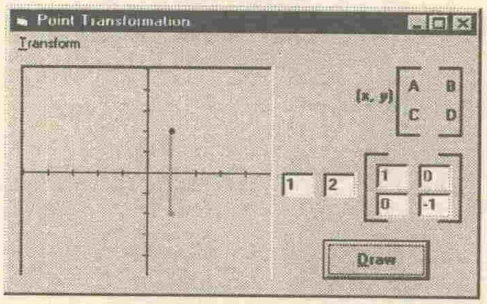
- คำสั่ง Scaling สำหรับการย้ายตำแหน่งของจุดไปยังตำแหน่งอื่น ๆ บนระนาบ x-y โดยที่ระยะที่จะเคลื่อนย้ายไปนั้น ก็ขึ้นกับค่าของสมาชิกของเมตริกซ์ T (transform matrix)



รูปที่ 2 แสดงผลลัพธ์ของคำสั่ง Scaling สำหรับจุด (1, 2)

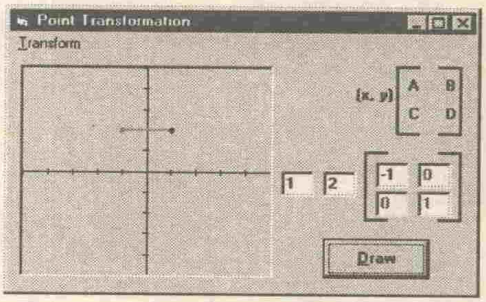
- คำสั่ง Reflection สำหรับการสะท้อนจุดกับแกน x หรือ y หรือจุดกำเนิด (0, 0) ในแนวตรงข้าม โดยที่ระยะห่างระหว่างตำแหน่งของจุดก่อนการสะท้อนจากแกนหรือจุดกำเนิด (0, 0) จะมีค่าเท่ากับระยะห่างระหว่างตำแหน่งของจุดหลังการสะท้อน ซึ่งประกอบด้วย 3 คำสั่งย่อย ดังนี้

X สำหรับการ reflection กับแกน x เป็นการสะท้อนตำแหน่งของจุดกับแกน x หรือย้ายจุดมายังฝั่งตรงข้ามของแกน x (หมายถึง การเปลี่ยนค่า y จากค่าบวกมาเป็นค่าลบหรือตรงกันข้าม)



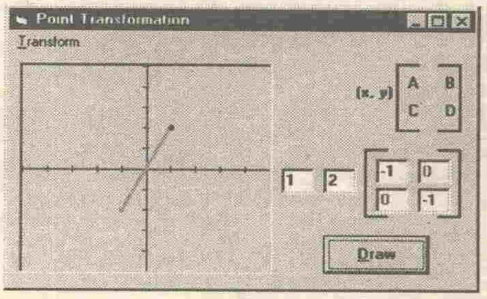
รูปที่ 3 แสดงผลลัพธ์ของคำสั่งย่อย X ของคำสั่ง Reflection สำหรับจุด (1, 2)

Y สำหรับการ reflection กับแกน y เป็นการสะท้อนตำแหน่งของจุดกับแกน y หรือย้ายจุดมายังฝั่งตรงข้ามของแกน y (หมายถึง การเปลี่ยนค่า x จากค่าบวกมาเป็นค่าลบ หรือตรงกันข้าม)



รูปที่ 4 แสดงผลลัพธ์ของคำสั่งย่อย Y ของคำสั่ง Reflection สำหรับจุด (1, 2)

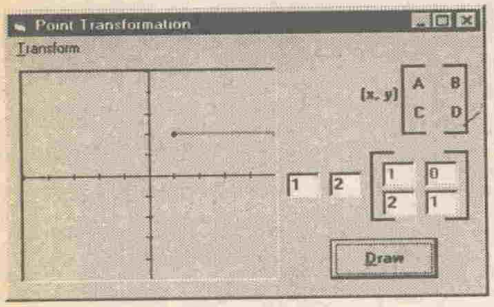
X and Y สำหรับการ reflection กับจุดกำเนิดเป็นการสะท้อนตำแหน่งของจุดกับแกน x และแกน y หรือย้ายจุดมายังฝั่งตรงข้ามของทั้งแกน x และแกน y (หมายถึง การเปลี่ยนค่า x และ y จากค่าบวกมาเป็นค่าลบ หรือตรงกันข้าม)



รูปที่ 5 แสดงผลลัพธ์ของคำสั่งย่อย X and Y ของคำสั่ง Reflection สำหรับจุด (1, 2)

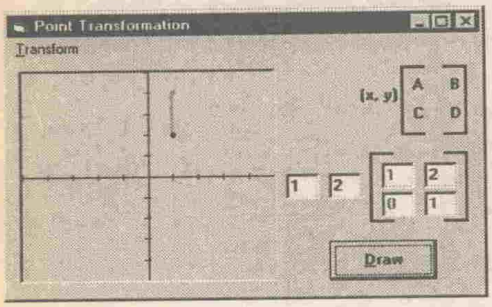
- คำสั่ง Shear สำหรับการย้ายจุดไปตามแกน x หรือ y หรือระนาบ x-y ด้วยระยะทางที่เป็นจำนวนเท่าของค่า y, x และ y, x ตามลำดับ ผู้อ่านจะเห็นว่าในความเป็นจริงสำหรับกรณีของจุด shear จะให้ผลเช่นเดียวกับการสะท้อน (reflection) เพียงแต่สำหรับในกรณีของการ shear นั้น จะเป็นการใช้ค่าของ x ในการคำนวณระยะทางการย้ายไปตามแกน y และในทางกลับกันจะเป็นการใช้ค่าของ y ในการคำนวณระยะทางการย้ายไปตามแกน x ซึ่งประกอบด้วย 3 คำสั่งย่อย ดังนี้

X สำหรับการ shear ตามแนวแกน x เป็นการย้ายจุดไปตามแกน x ด้วยระยะทางที่เป็นจำนวนเท่าของค่า y โดยที่ค่า y ของจุดไม่เปลี่ยนแปลง



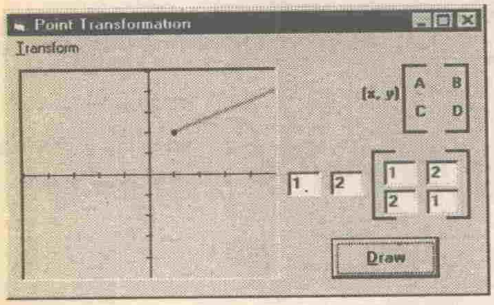
รูปที่ 6 แสดงผลลัพธ์ของคำสั่งย่อย X ของคำสั่ง Shear สำหรับจุด (1, 2)

Y สำหรับการ shear ตามแนวแกน y เป็นการย้ายจุดไปตามแกน y ด้วยระยะทางที่เป็นจำนวนเท่าของค่า x โดยที่ค่า x ของจุดไม่เปลี่ยนแปลง



รูปที่ 7 แสดงผลลัพธ์ของคำสั่งย่อย Y ของคำสั่ง Shear สำหรับจุด (1, 2)

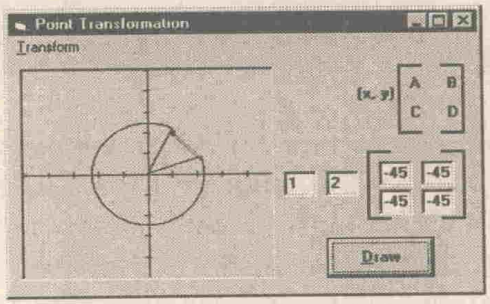
X and Y สำหรับการ shear ตามแนวระนาบ x-y เป็นการย้ายจุดไปตามแกน x และแกน y ด้วยระยะทางที่เป็นจำนวนเท่าของค่า y และจำนวนเท่าของค่า x ตามลำดับ



รูปที่ 8 แสดงผลลัพธ์ของคำสั่งย่อย X and Y ของคำสั่ง Shear สำหรับจุด (1, 2)

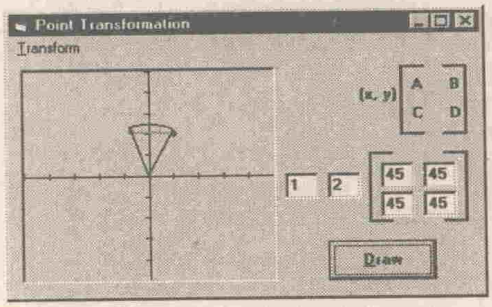
- คำสั่ง Rotation สำหรับการหมุนจุดซึ่งสามารถแบ่งออกได้เป็น 2 ทิศทาง คือ การหมุนจุดตามเข็มนาฬิกาและการหมุนจุดทวนเข็มนาฬิกา ซึ่งแนวคิดของการหมุนจุดก็มาจากแนวคิดจากพีชคณิตด้านการใช้ฟังก์ชันด้านตรีโกณมิติ sine หรือ cosine ร่วมกับการกำหนดตำแหน่งของคู่ลำดับในระนาบ x-y ด้วยระบบโพลาร์ (polar coordinate) ซึ่งประกอบด้วย 2 คำสั่งย่อย ดังนี้

Clockwise สำหรับการ rotation จุดตามเข็มนาฬิกา เป็นการหมุนจุดรอบจุดกำเนิด (0, 0) ในทิศทางตามเข็มนาฬิกา



รูปที่ 9 แสดงผลลัพธ์ของคำสั่งย่อย Clockwise ของคำสั่ง Rotation สำหรับจุด (1, 2)

Counter Clockwise สำหรับการ rotation จุดทวนเข็มนาฬิกา เป็นการหมุนจุดรอบจุดกำเนิด (0, 0) ในทิศทางทวนเข็มนาฬิกา



รูปที่ 10 แสดงผลลัพธ์ของคำสั่งย่อย Counter Clockwise ของคำสั่ง Rotation สำหรับจุด (1, 2)

การทำงานของโปรแกรม TransPnt

ก่อนที่ผู้เขียนจะอธิบายถึงการทำงานของโปรแกรมอย่างละเอียด ก็จะขออธิบายถึงค่าคงที่หรือตัวแปรที่สำคัญของโปรแกรมตัวอย่างเสียก่อนดังต่อไปนี้ สำหรับภายในโมดูล Trans2D.bas ก็จะประกอบด้วยค่าคงที่หรือตัวแปรที่สำคัญ ดังนี้

ค่าคงที่

```
Public Const PI As Double = 3.141592654
```

ค่าคงที่ PI (Pi) เป็นค่าคงที่ทางตรีโกณมิติเกี่ยวกับวงกลม โดยเป็นค่าที่ได้จากอัตราส่วนระหว่างความยาวของเส้นรอบวงกลมหารด้วยค่าความยาวเส้นผ่านศูนย์กลางวงกลม ซึ่งจะมีค่าคงที่สำหรับวงกลมทุกขนาด

ตัวแปรชนิดข้อมูลโครงสร้าง

```
Type CoXYType
    xCo     As Double
    yCo     As Double
End Type
Public CoXY As CoXYType
```

ตัวแปร CoXY เป็นตัวแปรชนิดข้อมูลโครงสร้าง CoXYType ซึ่งประกอบด้วย 2 สมาชิก xCo สำหรับจัดเก็บค่า x ของคู่ลำดับ (x, y) และ yCo สำหรับจัดเก็บค่า y ของคู่ลำดับ (x, y) ตามลำดับ

```
Type Matrix2DType
    r1c1    As Double
    r1c2    As Double
    r2c1    As Double
    r2c2    As Double
End Type
Public Matrix2D As Matrix2DType
```

ตัวแปร Matrix2D เป็นตัวแปรชนิดข้อมูลโครงสร้าง Matrix2DType ซึ่งประกอบด้วย 4 สมาชิก r1c1, r1c2, r2c1 และ r2c2 สำหรับจัดเก็บค่าของสมาชิก a_{11}, a_{12}, a_{21} และ a_{22} ของเมตริกซ์ T ตามลำดับ

สำหรับภายในฟอร์ม TransPnt.frm ก็จะประกอบด้วยค่าคงที่หรือตัวแปรที่สำคัญ ดังนี้

ค่าคงที่

```
Private Const TickSize As Single = 0.15
```

ค่าคงที่สำหรับกำหนดขนาดความยาวของขีดแบ่งสเกลบนแกน x-y ซึ่งผู้เขียนได้กำหนดให้มีค่าเท่ากับ 0.15 ซ.ม. ถ้าหากผู้อ่านเห็นว่าเป็นค่าที่ไม่เหมาะสม ก็สามารถทำการแก้ไขได้

ตัวแปร

```
Private CoXYtmp As CoXYType
```

ตัวแปร CoXYtmp เป็นตัวแปรชนิดข้อมูลโครงสร้าง CoXYType ซึ่งผู้เขียนใช้จัดเก็บตำแหน่งของคู่ลำดับก่อนที่จะถูก transformation ทั้งนี้เพื่อใช้ในการแสดงผลจุดภายหลังจากที่มีการคลิกปุ่มคำสั่ง Draw

```
Private Matrix2Dtmp As Matrix2DType
```

ตัวแปร Matrix2Dtmp เป็นตัวแปรชนิดข้อมูลโครงสร้าง Matrix2DType ซึ่งผู้เขียนใช้จัดเก็บค่าของ Matrix2D ในกรณีที่มีการเลือกคำสั่ง Rotation (ตัวแปร RotateFlag มีค่าเท่ากับ True)

```
Private RotateFlag As Boolean
```

ตัวแปร RotateFlag ใช้ในการแสดงในกรณีที่มีการเลือกคำสั่ง Rotation

เมื่อผู้อ่านรันโปรแกรมตัวอย่างโพรซีเดอร์เหตุการณ์ Form_Load จะถูกเรียก ซึ่งภายในโพรซีเดอร์เหตุการณ์นี้ผู้เขียนจะกำหนดให้ตำแหน่งคู่ลำดับ (x, y) ของจุดที่ใช้ในการ transformation มีค่าเท่ากับ (1, 2) ทั้งสำหรับตัวแปร CoXYtmp และ CoXY ดังนี้

```
CoXYtmp.xCo = 1
CoXYtmp.yCo = 2
CoXY.xCo = 1
CoXY.yCo = 2
```

และในขณะเดียวกันผู้เขียนก็กำหนดให้มีการเรียกใช้โพรซีเดอร์ DrawAxis ภายในโพรซีเดอร์เหตุการณ์ Form_Activate ทั้งนี้เพื่อให้มีการวาดแกน x-y และเส้นแบ่งสเกลบนแกนทั้งสองทุกครั้งที่ Visual Basic มีความต้องการวาดพื้นที่แสดงผลของฟอร์มใหม่ทุกครั้ง สำหรับโค้ดภายในโพรซีเดอร์ DrawAxis จะเริ่มต้นด้วยการลบพื้นที่แสดงผลของคอนโทรล Picture1 และทำการวาดเส้นแกน x-y ด้วยเส้นทึบสีแดง ดังนี้

```
Picture1.Cls
Picture1.DrawStyle = vbSolid
' draw X-axis.
Picture1.Line (Picture1.ScaleLeft, 0)-(Picture1.ScaleWidth, 0), vbRed
' draw Y-axis.
Picture1.Line (0, Picture1.ScaleTop)-(0, Picture1.ScaleHeight), vbRed
Picture1.DrawStyle = vbSolid
```

จากนั้นก็ทำการวาดเส้นแบ่งสเกลบนแกนทั้งสองโดยที่โค้ด Picture1.Line แต่ละบรรทัดจะทำการวาดเส้นดังกล่าวบนแกน x, -x, y และ –y ตามลำดับ ด้วยเส้นทึบสีแดง

```
For i = 1 To 5
   Picture1.Line (i, -TickSize)-(i, TickSize), vbRed
   Picture1.Line (-i, -TickSize)-(-i, TickSize), vbRed
   Picture1.Line (-TickSize, i)-(TickSize, i), vbRed
   Picture1.Line (-TickSize, -i)-(TickSize, -i), vbRed
Next
```

เมื่อสิ้นสุดขั้นตอนการเริ่มต้นของโปรแกรมแล้ว จากนั้นโปรแกรมก็จะรอให้ผู้อ่านเลือกคำสั่งจากเมนู Transform และคลิกที่ปุ่มคำสั่ง Draw โดยที่แต่ละคำสั่งจะถูกควบคุมด้วยโพรซีเดอร์เหตุการณ์ ดังต่อไปนี้

เมื่อเลือกคำสั่ง Scaling โพรซีเดอร์เหตุการณ์ mnuTransScaling_Click ก็จะถูกเรียก ซึ่งภายในโพรซีเดอร์เหตุการณ์นี้ผู้เขียนก็จะทำการกำหนดค่าของสมาชิกของเมตริกซ์ทั้ง 4 สมาชิก โดยจะกำหนดให้มีค่าตามที่เคยกล่าวมาแล้วในบทความตอนที่แล้ว จากนั้นก็เรียกโพรซีเดอร์ UpdateMatrix เพื่อทำการแสดงผลค่าของทุกสมาชิกภายในเมตริกซ์ดังที่ปรากฏบนฟอร์มของโปรแกรม และกำหนดให้ตัวแปร RotateFlag มีค่าเท่ากับ False เพื่อแสดงว่าคำสั่งดังกล่าวไม่ใช่คำสั่งย่อยของเมนู Rotation

```
Matrix2D.r1c1 = 2
Matrix2D.r1c2 = 0
Matrix2D.r2c1 = 0
Matrix2D.r2c2 = 2
UpdateMatrix
RotateFlag = False
```

โดยที่โพรซีเดอร์ UpdateMatrix จะทำการกำหนดค่าคู่ลำดับ (ในที่นี้หมายถึงคู่ลำดับ (1, 2)) จากตัวแปรโครงสร้าง CoXYtmp ให้กับ CoXY และกำหนดค่า x และ y ให้กับคอนโทรลอะเรย์ Text1 สมาชิกลำดับ 0 และ 1 ตามลำดับ

```
CoXY.xCo = CoXYtmp.xCo
CoXY.yCo = CoXYtmp.yCo
Text1(0) = CoXY.xCo
Text1(1) = CoXY.yCo
```

จากนั้นก็กำหนดค่าสมาชิก r1c1, r1c2, r2c1 และ r2c2 ให้กับคอนโทรลอะเรย์ Text3 ทั้ง 4 ตัว ตามลำดับ และถ้าหากสมาชิกลำดับใดที่มีค่าเท่ากับ 0 หรือ 1 ก็กำหนดให้คอนโทรลอะเรย์ Text3 ดังกล่าวมีสถานะเป็น disabled ซึ่งจะไม่สามารถแก้ไขค่าในคอนโทรลอะเรย์ Text3 นั้น ๆ และสุดท้ายก็เรียกโพรซีเดอร์ DrawPoint เพื่อวาดจุดบนคอนโทรล Picture1

```
Text3(0) = Matrix2D.r1c1
Text3(0).Enabled = (Matrix2D.r1c1 <> 0) And (Matrix2D.r1c1 <> 1)
Text3(1) = Matrix2D.r1c2
Text3(1).Enabled = (Matrix2D.r1c2 <> 0) And (Matrix2D.r1c2 <> 1)
Text3(2) = Matrix2D.r2c1
Text3(2).Enabled = (Matrix2D.r2c1 <> 0) And (Matrix2D.r2c1 <> 1)
Text3(3) = Matrix2D.r2c2
Text3(3).Enabled = (Matrix2D.r2c2 <> 0) And (Matrix2D.r2c2 <> 1)

DrawPoint
```

สำหรับโพรซีเดอร์ DrawPoint จะเริ่มต้นด้วยการตรวจสอบว่าข้อมูลที่ปรากฏในคอนโทรลอะเรย์ Text1 นั้นเป็นค่าตัวเลขหรือไม่โดยใช้ฟังก์ชัน IsNumeric ซึ่งถ้าหากเป็นค่าตัวเลขก็จะทำการวาดแกน x-y โดยการเรียกใช้โพรซีเดอร์ DrawAxis จากนั้นก็ทำการวาดจุดตรงตำแหน่งที่กำหนด ซึ่งในที่นี้ผู้เขียนจะแสดงผลจุดด้วยการวาดวงกลมสีเขียวที่มีรัศมีเท่ากับ 0.1 ซ.ม.โดยมีจุดศูนย์กลางที่ตำแหน่ง (x, y) ที่กำหนดในคอนโทรลอะเรย์ Text1 ดังนี้

```
If IsNumeric(Text1(0)) And IsNumeric(Text1(1)) Then
    DrawAxis

    Picture1.DrawWidth = 1
    Picture1.DrawStyle = vbSolid
    Picture1.FillStyle = vbFSSolid
    Picture1.FillColor = vbGreen
    ' update arbitary point if coordinate value
     ' is revised.
    Picture1.Circle (Text1(0), Text1(1)), 0.1, vbGreen
    Picture1.FillStyle = vbFSTransparent
End If
```

จากนั้นเมื่อผู้อ่านคลิกที่ปุ่มคำสั่ง Draw โพรซีเดอร์เหตุการณ์ Command1_Click ก็จะถูกเรียก ซึ่งภายในโพรซีเดอร์เหตุการณ์นี้จะมีการเรียกโพรซีเดอร์ UpdateValue เพื่อปรับปรุงค่าในเมตริกซ์ลงในตัวแปรสำหรับใช้ในการคำนวณต่อไป ซึ่งภายในโพรซีเดอร์ UpdateValue ก็จะเริ่มต้นด้วยการกำหนดค่าจากคอนโทรลอะเรย์ Text1 ให้กับตัวแปร CoXY จากนั้นก็กำหนดค่าจากคอนโทรลอะเรย์ Text3 ให้กับตัวแปร Matrix2D และทำการวาดแกน x-y โดยการเรียกโพรซีเดอร์ DrawAxis ดังนี้

```
CoXY.xCo = Text1(0)
CoXY.yCo = Text1(1)
CoXYtmp = CoXY

' update Text3 array to transform matrix.
Matrix2D.r1c1 = Text3(0)
Matrix2D.r1c2 = Text3(1)
Matrix2D.r2c1 = Text3(2)
Matrix2D.r2c2 = Text3(3)

DrawAxis
```

จากนั้นก็ทำการตรวจสอบว่าคำสั่งที่ถูกเลือกเป็นคำสั่งย่อยของเมนู Rotation หรือไม่ โดยการตรวจสอบจากค่าตัวแปร RotateFlag ซึ่งถ้าหากใช่ ก็ให้ทำการจัดเก็บค่าของเมตริกซ์ลงในตัวแปร Matrix2Dtmp จากนั้นก็เรียกโพรซีเดอร์ TransRotationC2D เพื่อทำการคำนวณและหมุนจุดตามที่กำหนด แต่ถ้าหากไม่ใช่ก็จะเรียกโพรซีเดอร์ CalMatrix เพื่อทำการคูณเมตริกซ์ T เข้ากับคู่ลำดับ (x, y) เพื่อทำการ transform จุดดังนี้

```
.If RotateFlag Then
    Matrix2Dtmp = Matrix2D
    TransRotationC2D
 Else
    CalMatrix
 End If
```

สุดท้ายเมื่อตำแหน่งใหม่ของคู่ลำดับถูกคำนวณเรียบร้อยแล้ว ก็ให้ทำการวาดจุดโดยการเรียกใช้โพรซีเดอร์ DrawObj จากนั้นถ้าหากเป็นกรณีของตัวแปร RotateFlag มีค่าเท่ากับ True ก็ให้คืนกลับค่าของเมตริกซ์ให้กับตัวแปร Matrix2D

```
DrawObj
 If RotateFlag Then
    Matrix2D = Matrix2Dtmp
 End If
```

สำหรับโพรซีเดอร์ CalMatrix ก็เป็นเมตริกซ์ง่าย ๆ ที่ทำการคูณเมตริกซ์ T เข้ากับคู่ลำดับ (x, y) และกำหนดค่าหลังการคำนวณให้กับตัวแปร CoXY ดังนี้

```
Dim CoXYcalc As CoXYType

CoXYcalc.xCo = Matrix2D.r1c1 * CoXY.xCo + Matrix2D.r2c1 * CoXY.yCo
CoXYcalc.yCo = Matrix2D.r1c2 * CoXY.xCo + Matrix2D.r2c2 * CoXY.yCo
CoXY = CoXYcalc
```

สำหรับโพรซีเดอร์ TransRotationC2D เป็นโพรซีเดอร์พิเศษสำหรับกรณีของการหมุนจุดเท่านั้น โดยภายในโพรซีเดอร์นี้ จะเป็นการแก้ไขค่าของสมาชิกจากค่าตัวเลขของศา เป็นค่าของฟังก์ชัน cosine และ sine โดยใช้ฟังก์ชัน Cos และ Sin ของ Visual Basic ตามลำดับ สำหรับนิพจน์ภายในฟังก์ชันทั้งสองนั้น ก็เป็นการแปลงจากหน่วยของมุมจากหน่วยของศามาเป็นหน่วยเรเดียน (180 องศา มีค่าเท่ากับ π เรเดียน) ทั้งนี้เนื่องจากฟังก์ชันทั้งสองจะคำนวณโดยการกำหนดให้มุมในหน่วยเรเดียนเท่านั้น จากนั้นแล้วจึงเรียกโพรซีเดอร์ CalMatrix เพื่อคำนวณการคูณเมตริกซ์ต่อไป

```
Matrix2D.r1c1 = Cos(Matrix2D.r1c1 * PI / 180)
Matrix2D.r1c2 = Sin(Matrix2D.r1c2 * PI / 180)
Matrix2D.r2c1 = -Sin(Matrix2D.r2c1 * PI / 180)
Matrix2D.r2c2 = Cos(Matrix2D.r2c2 * PI / 180)
CalMatrix
```

สำหรับโพรซีเดอร์ DrawObj วาดเส้นเชื่อมระหว่างจุดก่อนและหลังการ transform ซึ่งถูกจัดเก็บในตัวแปร CoXYtmp และ CoXYtmp ตามลำดับ

```
Picture1.DrawStyle = vbDot
Picture1.DrawWidth = 2
' draw line between points.
Picture1.Line (CoXYtmp.xCo, CoXYtmp.yCo)-(CoXY.xCo, CoXY.yCo), vbYellow
```

จากนั้นก็วาดจุดโดยผู้เขียนกำหนดให้มีการวาดกลมรัศมี 0.1 ซ.ม. ล้อมรอบจุดดังกล่าวด้วยสีเขียวและสีดำตามลำดับ

```
Picture1.DrawWidth = 1
Picture1.DrawStyle = vbSolid
Picture1.FillStyle = vbFSSolid
Picture1.FillColor = vbGreen
' draw transformed point.
Picture1.Circle (CoXY.xCo, CoXY.yCo), 0.1, vbGreen
Picture1.FillColor = vbBlack
' draw original point.
Picture1.Circle (CoXYtmp.xCo, CoXYtmp.yCo), 0.1, vbBlack
Picture1.FillStyle = vbFSTransparent
```

สำหรับในขั้นตอนสุดท้าย ก็จะเป็นการวาดส่วนโค้งของวงกลม (สำหรับในกรณีของคำสั่ง Rotation เท่านั้น) เชื่อมจุดทั้งสอง ทั้งนี้เพื่อต้องการแสดงให้ผู้อ่านเห็นว่าการหมุนจุดนั้น ก็หมายถึงการเคลื่อนย้ายจุดไปยังตำแหน่งต่าง ๆ บนส่วนโค้งวงกลมนั่นเอง โดยในที่นี้ผู้เขียนจะกำหนดให้ลากเส้นเชื่อมจุดทั้งสองด้วยส่วนโค้งของวงกลมที่สั้นกว่าสำหรับในกรณีของการหมุนจุดแบบทวนเข็มนาฬิกา (CCW) และในทางกลับกันให้ลากเส้นเชื่อมจุดทั้งสองด้วยส่วนโค้งของวงกลมที่ยาวกว่าสำหรับในกรณีของการหมุนจุดแบบตามเข็มนาฬิกา (CCW) แต่เนื่องจากการวาดส่วนโค้งของวงกลมด้วยประโยค Circle นั้น มีข้อจำกัดในเรื่องของการกำหนดตำแหน่งมุมเริ่มต้นและมุมสิ้นสุดของส่วนโค้งของวงกลม สำหรับในแต่ละ quardant ของระนาบ x-y ที่แตกต่างกัน ดังนั้นจึงต้องการแปลงมุมให้ตรงกับข้อกำหนดดังกล่าวข้างต้น ซึ่งรายละเอียดที่เหลือผู้อ่านสามารถศึกษาได้จากหนังสือ การใช้โปรแกรม Visual Basic 5.0 Professional Edition ซึ่งแต่งโดยตัวผู้เขียน จัดพิมพ์และจำหน่ายโดยบริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) ครับ

```
If RotateFlag Then
    Dim radius As Double
    Dim StartArc As Double, EndArc As Double

    radius = Sqr(CoXYtmp.xCo ^ 2 + CoXYtmp.yCo ^ 2)
    StartArc = Atn(CoXYtmp.yCo / CoXYtmp.xCo)
    EndArc = Atn(CoXY.yCo / CoXY.xCo)
    If CoXYtmp.xCo < 0 And CoXYtmp.yCo > 0 Then
        StartArc = PI + StartArc
    ElseIf CoXYtmp.xCo < 0 And CoXYtmp.yCo < 0 Then
        StartArc = StartArc + PI
    ElseIf CoXYtmp.xCo > 0 And CoXYtmp.yCo < 0 Then
        StartArc = 2 * PI + StartArc
     End If
    If CoXY.xCo < 0 And CoXY.yCo > 0 Then
        EndArc = PI + EndArc
    ElseIf CoXY.xCo < 0 And CoXY.yCo < 0 Then
        EndArc = EndArc + PI
    ElseIf CoXY.xCo > 0 And CoXY.yCo < 0 Then
        EndArc = 2 * PI + EndArc
     End If
   Picture1.Circle (0, 0), radius, vbBlack, -StartArc, -EndArc
End If
```

สำหรับในส่วนของคำสั่งอื่น ๆ ของโปรแกรมตัวอย่าง ก็จะมีการเขียนโค้ดที่แตกต่างเฉพาะในส่วนของการกำหนดค่าสมาชิกของเมตริกซ์ T เท่านั้น ซึ่งผู้เขียนคิดว่าผู้อ่านสามารถที่จะศึกษาเองได้ จึงขออนุญาตไม่ขอกล่าวซ้ำในที่นี้ และในส่วนท้ายของบทความก็จะแสดงซอร์สโค้ดของโปรเจ็กต์ TransPnt.vbp และสำหรับโค้ดในไฟล์โมดูล Trans2D.bas นั้นที่ผู้เขียนไม่กล่าวถึง เพราะโดยส่วนใหญ่เป็นโพรซีเดอร์ที่ช่วยในการกำหนดค่าสมาชิกของเมตริกซ์ T สำหรับการ transformation ในกรณีต่าง ๆ เท่านั้น ทั้งนี้เพื่อให้ง่ายต่อการนำไปใช้โดยไม่ต้องจดจำค่าสมาชิกของเมตริกซ์ T สำหรับในแต่ละกรณี เช่น ในกรณีของการสะท้อนจุด (reflection) กับแกน x ผู้อ่านก็เพียงแต่เขียนโค้ดเพื่อกำหนดค่าคู่ลำดับ (x, y) ให้กับตัวแปร CoXY จากนั้นก็สามารถเรียกใช้โพรซีเดอร์ TransReflectionX2D เพื่อคำนวณตำแหน่งใหม่ของคู่ลำดับ (x', y') ได้ทันที ซึ่งจะเห็นว่าสามารถลดความยุ่งยากลงไปได้มากทีเดียว

```
Sub TransReflectionX2D()
    Matrix2D.r1c1 = -1
    Matrix2D.r1c2 = 0
    Matrix2D.r2c1 = 0
    Matrix2D.r2c2 = 1
    CalMatrix
End Sub
```

สำหรับโพรซีเดอร์อื่น ๆ ที่เหลือขอให้ศึกษาเองนะครับ ถ้าติดขัดประการใดก็ e-mail มาสอบถามกันได้ ส่วนในตอนต่อไปจะกล่าวถึงการ transformation ออบเจ็กต์ในระนาบ x-y ที่มีความซับซ้อนมากขึ้น โดยจะใช้รูปสามเหลี่ยมเป็นตัวแทนของออบเจ็กต์ 2 มิติ โปรดติดตามต่อไปครับ

ลิสติงที่ 1
ไฟล์โปรเจ็กต์ TransPnt.vbp

```
Type=Exe
Form=TransPnt.frm
Module=basTrans2D; Trans2D.bas
IconForm="Form1"
Startup="Form1"
HelpFile=""
Title="Point Transformation"
ExeName32="TransPnt.exe"
Path32="C:\"
Command32=""
Name="PointTrans"
HelpContextID="0"
CompatibleMode="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionComments="Multimedia API Programming by VB/Win"
VersionCompanyName="Microcomputer Magazine SE-ED"
VersionFileDescription="2-D Point Transformation Program"
VersionLegalCopyright="Sutthisak Phongthanapanich"
CompilationType=0
OptimizationType=0
FavorPentiumPro(tm)=0
CodeViewDebugInfo=0
NoAliasing=0
BoundsCheck=0
OverflowCheck=0
FlPointCheck=0
FDIVCheck=-1
UnroundedFP=0
StartMode=0
Unattended=0
ThreadPerObject=0
MaxNumberOfThreads=1
```

เทคนิคประจำคอลัมน์

ถาม วินโดวส์ 95 จัดเก็บข้อมูลการลงทะเบียนของโปรแกรมทั้งหมดไว้ตรงไหน?

ตอบ ข้อมูลการลงทะเบียนของโปรแกรมทั้งหมดจะถูกวินโดวส์ 95 จัดเก็บลงในไฟล์ registry ซึ่งจะมีอยู่ด้วยกัน 2 ไฟล์ คือ System.dat และ User.dat ในไดเรคทอรี \Windows นั่นเอง ซึ่งผู้อ่านสามารถเปิดอ่านข้อมูลในไฟล์ registry ดังกล่าวได้ โดยใช้โปรแกรม Regedit.exe ของวินโดวส์ครับ